Web Services
A Three(?) Part Tutorial

Part II
Reprise, Tomcat, and (maybe) SOAP

First – in Response to some Suggestions and Questions

- Java in 5 minutes
- Lexical Analysis vs Compilation and XML
- An XML walker
- Pruning an XML tree

Java in 5 minutes (for C++ programmers)

- Almost any executable line of Java code will be understandable to C++ programmers.
  - Arithmetic – same (but data in Java is uniform)
  - Flow control – same (but Java programmers tend to use try/catch exception handling more)
  - function invocation – same
- What's missing?
  - * and -> pointer dereferencing
  - Everything in Java is within a class

What is missing in Java?

- No operator overloading
- No template library (but generic classes are expected, perhaps within a year - “tiger” release, Java 1.5)

What Features does Java Provide?

- Memory management – no more calls to delete
- Built in threading – every object is-a – *monitor*
  - Threads can wait on and signal any object
  - Code can be synchronized at the block or method level
- A single inheritance tree, from Object
  - all methods are “virtual”
  - Interfaces used as a lightweight (but design friendly) multiple inheritance.
- A huge standard library

Details

- Java Virtual Machine makes programs machine and O/S independent
- Very clever JVMs provide run-time optimization
- The run time stack contains only primative data types and references (very smart pointers)
- All dynamic memory (via new) is in the “heap” which is an object – that is, it protects itself from malicious and stupid programmers.

Basic Form

- Generally, 1 class is in 1 and only 1 file. (Files can, but seldom do, have more than one class.)
- Every class has a “main” method – the JVM is told which class to run.
  - Most main methods can be used for regression testing of the class
- Every class has (via Object) an equals(), toString(), and hashCode() method.
- Get/Set semantics makes “Beans” amenable to automatic programming.

A Simple Class

```
public class Circle {

   private double radius;

   public Circle() {
      this(1.0);
   }

   public Circle(double r) {
      setRadius(r);
   }

   public void setRadius(double r) {
      assert r >= 0.0 : "radius must be >= 0.0";
      radius = r;
   }

   public double getRadius() {
      return radius;
   }

   public double getArea() {
      return Math.PI*radius*radius;
   }

   public boolean equals(Object o) {
      return radius == ((Circle)o).getRadius();
   }

   public String toString() {
      return "Circle with radius " + radius;
   }

   public static void main(String[] args) {
      Circle c1 = new Circle(1.0);
      Circle c2 = new Circle();
      System.out.println(c1 + " has area " + c1.getArea());
      if (c1.equals(c2))
         System.out.println(c1 + " " + c2 + " are equal");
      Circle c3 = new Circle(-4.0);
   }
}
```

What Drives People Nuts

- package/import
  - package is a mix of namespace and directory – e.g. “package gov.bnl.Rhic”
    - Somewhere, there is a directory tree .../gov/bnl/Rhic/
    - When you are compiling or running, you must know where this tree is (CLASSPATH environment)
  - import is not include
    - It does tell the compiler what packages to look at (for signatures etc.) when an unknown name appears.
    - You can always use fully qualified names instead

Why You Should Consider Java?

- Fast enough, probably faster on a lifetime basis
- Much easier than C++ (you don't have to be nearly so bright to program well with Java)
- Truly is portable
- If you need C++, you can use it (JNI, Corba, etc.)
- If not Java, then at least consider the JVM
  - virtually any language will run on the JVM
- It drives Bill Gates nuts.

Questions from Last Week

- “I don't get the difference between SAX and DOM”
  - Traditional compilers have (at least) 2 phases
    - Lexical analysis
      - type, identifier, open paren, identifier, comma, identifier, close paren, open curly, close curly
    - Compilation – tree building
      - Compilation Unit
        - function declaration
          - function header/function body
- In this view, SAX is the lexical analysis, DOM is the compilation. On occasion, SAX is enough.

SAX Classes

- SAX makes use of a number of classes
  - ContentHandler – this is an interface that defines 10 or so “call-back” methods. When the XML document is being read, every important lexical junk generates a call to one of these methods
  - XMLReader – this is the scanner than reads the XML data and generates the proper calls to the Content Handler
  - InputSource – wraps a byte/character source for the XML data
- Example – has to do with the next question

Question 2 – Pruning an XML Document

- DOM stores the entire XML document in memory
  - This may be too demanding of space and time resources
  - One may know in advance not to follow certain branches of the XML “tree”.
- SAX scans the XML linearly and is in an ideal position to ignore large chunks.
- DOM gives a better representation in memory
- So – why not use both?

Example 1 – Prune out the Authors and all inside the Author Element

- SAX will provide us with each XML token
- Pruner will look for any number of elements
  - If one of the elements to be pruned is found, it will start skipping
  - When the skipping element ends, it will start processing again.
- Anything not pruned is written to some output stream.

Recall, the XML Document

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : DesignPatterns.xml
    Created on : September 7, 2003, 12:52 PM
    Author     : Dave Stampf
    Description:
        Purpose of the document follows.
-->

<book isbn="0-201-633511-2">
    <title>Design Patterns</title>
    <subtitle>Elements of Reusable Object-Oriented Software</subtitle>
    <author>Eric Gamma</author>
    <author>Richard Helm</author>
    <author>Ralph Johnson</author>
    <author>John Vlissides</author>
    <publisher>Addison-Wesley Publishing Company</publisher>
    <copyright>1995</copyright>
    <hardcover />

</book>
```

Pruning with SAX – 1

```
import java.util.*;
import java.io.*;
import javax.xml.parsers.*;
import org.xml.sax.*;


public class Pruner implements ContentHandler {

    private Vector pruneAt;     // elements to prune
    private PrintStream ps;     // where the output goes
    private boolean skip = false;   // skipping pruned parts
    private String skipTo = ""; // what started the pruning

    /** Creates a new instance of Pruner */
    public Pruner() {
        pruneAt = new Vector();
    }

    public void addPrunePoint(String s) {
        pruneAt.add(s);
    }

    public void setOutputStream(OutputStream os) {
        ps = new PrintStream(os);
    }

    public void startDocument() throws SAXException {
        ps.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
    }


    public void characters(char[] ch, int start, int length) throws SAXException {
        if (!skip) {    // if not skipping, copy characters
            for (int i = start; i < start+length; i++) {
                ps.print(ch[i]);
            }
        }
    }

    public void endDocument() throws SAXException {
        ps.close();
    }
        if (skip & skipTo.equals(qName)) {  // see if it is time to stop skip
            skip = false;
        } else {
            ps.println("\n</" + qName + ">");
        }
    }


    public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
throws SAXException {
        if (!skip) {    // if we aren't skipping, see if we should
            if (this.pruneAt.contains(qName)) {
                skip = true;
                skipTo = qName;
            } else {    // otherwise, ouput start of element
                ps.print("<" + qName + " ");
                if (atts != null) {
                    for (int i = 0; i < atts.getLength(); i++) {
                        ps.print(atts.getQName(i) + "=\"" + atts.getValue(i) + "\"");
                    }
                }
                ps.println(">");
            }
        }
    }

    public void startPrefixMapping(String prefix, String uri) throws SAXException {
    }

    public void endPrefixMapping(String prefix) throws SAXException {
    }

    public void ignorableWhitespace(char[] ch, int start, int length) throws SAXException {
    }

    public void processingInstruction(String target, String data) throws SAXException {
    }

    public void setDocumentLocator(Locator locator) {
    }

    public void skippedEntity(String name) throws SAXException {
    }
```

Pruning with SAX – 2

```
    // test the Prune content handler.

    public static void main(String[] args) throws Exception {

        Pruner p = new Pruner();
        p.setOutputStream(System.out);
        p.addPrunePoint("author");

        SAXParserFactory spf = SAXParserFactory.newInstance();
        SAXParser sp = spf.newSAXParser();
        XMLReader xr = sp.getXMLReader();

        xr.setContentHandler(p);

        Reader r = new FileReader("C:/WebServicesTutorial/DesignPatternsWithDTD.xml");
        InputSource is = new InputSource(r);
        xr.parse(is);
    }
}
```

And the Output...

```
<?xml version="1.0" encoding="UTF-8"?>
<book isbn="0-201-633511-2">
<title >
Design Patterns
</title>
<subtitle >
Elements of Reusable Object-Oriented Software
</subtitle>
<publisher >
Addison-Wesley Publishing Company
</publisher>
<copyright >
1995
</copyright>
<hardcover >
</hardcover>
</book>
```

This Doesn't Exactly Answer the Question!

- We don't have anything in memory to work with after this program runs – we really want an XML Document
- But, the old Unix concept of a pipeline can save the day – too bad Java is portable to systems that might not have Unix shells...
- The following example shows how it might be done internally to Java (using Threads as well!)

Pipelined XML Processing – 1

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.w3c.dom.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;

public class PrunedBook {

    private PipedOutputStream pos;
    private PipedInputStream pis;

    private Pruner pruner;
    private InputSource is;

    /** Creates a new instance of PrunedBook */
    public PrunedBook() throws Exception {

        // create a pipe for the two threads - the pruner and me

        pos = new PipedOutputStream();
        pis = new PipedInputStream(pos);

        // create the Pruner

        pruner = new Pruner();
        pruner.setOutputStream(pos);
        pruner.addPrunePoint("author");

        // access the XML file

        Reader r = new
FileReader("C:/WebServicesTutorial/DesignPatternsWithDTD.xml"
);
        is = new InputSource(r);

        // OK - let the pruner do its work in a separate thread.

        (new Thread(new Runnable () {
            public void run() {
                try {
                    SAXParserFactory spf =
SAXParserFactory.newInstance();
                    SAXParser sp = spf.newSAXParser();
                    XMLReader xr = sp.getXMLReader();

                    xr.setContentHandler(pruner);
                    xr.parse(is);
                } catch (Exception e) {
                }
            }
        })).start();

        // while the pruner is working, one can start up DOM
    }
```

Pipelined XML Processing – 2

```
  public void process() throws Exception {
     DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
     DocumentBuilder db = dbf.newDocumentBuilder();
     InputSource is = new InputSource(pis);

     Document doc = db.parse(is);

     // transform into output again

     TransformerFactory tf = TransformerFactory.newInstance();
     Transformer t = tf.newTransformer();
     t.transform(new DOMSource(doc), new StreamResult(System.out));
  }

  public static void main(String[] args) throws Exception {
    PrunedBook pb = new PrunedBook();
    pb.process();
  }
}
```

Remarks

- There are more elegant (but more complicated) ways to do the same thing using XMLFilter to handle the SAX processing, but one still has to make the transition to DOM
- There are some loose ends, but ...
- Notice how the factory pattern showed up 3 times

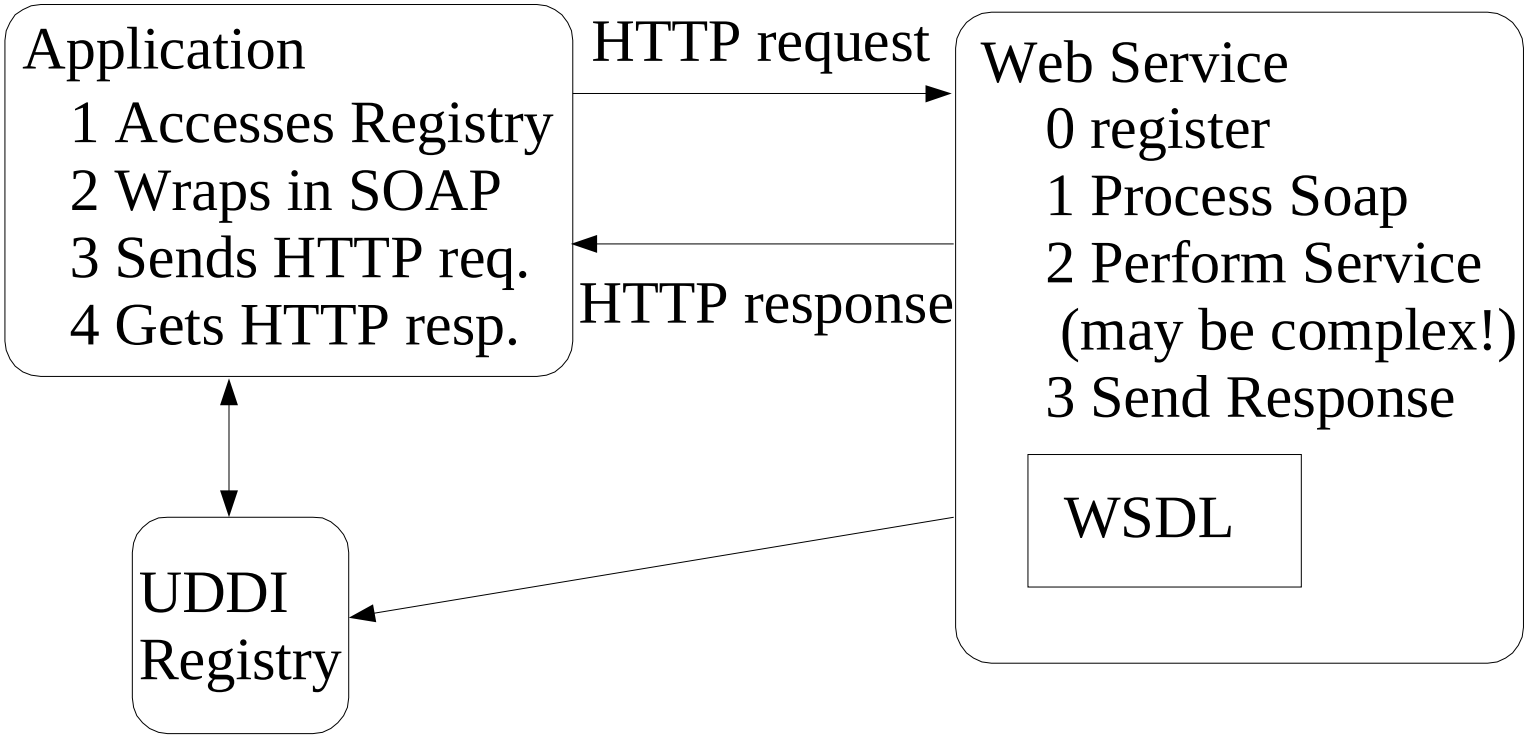
Recap

- Web Services
- XML
- XML Libraries

Web Services Definition

- A web service is a software system
  - identified by a URI
  - with public interfaces and bindings are defined and described using XML
  - whose definition can be discovered by other software systems.
  - and these other software systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet Protocols.

A Simple Architecture Example



XML – Extensible Markup Language

- One technique formatting data that provides
  - Structure
  - self-defining
  - machine independent
  - well formed/validity checks
  - standardized libraries for access in multiple languages
- Even Bill Gates likes it.

Homework

- Find the proper XML libraries for your favorite language, get them installed and create a Hello World XML file. (If Java, install the “jwsdp”)
- Write a program to “walk the tree” and pretty-print out the tags. I'll show you mine next time
- Find any data format application you have now and re-phrase it in terms of XML. Write a simple output method and some useful access methods. Then, stand back.

Walking the DOM Tree (everything is a Node)

```
import java.io.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import org.xml.sax.InputSource;
import org.w3c.dom.*;

public class Walk {

   public Walk(Reader r) throws Exception {
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
      dbf.setValidating(true);
      DocumentBuilder db = dbf.newDocumentBuilder();
      InputSource is = new InputSource(r);

      Document doc = db.parse(is);
      Node root = doc.getDocumentElement();
      walkTheDoc(root,0);
   }


   private void indent(int n) {
      while (n-- > 0) System.out.print(" ");
   }

   public static void main(String[] args) throws Exception{
      Reader r = new
          FileReader("C:/WebServicesTutorial/DesignPatternsWithDTD.xml");
      Walk b = new Walk(r);
   }

   private void walkTheDoc(Node n, int in) throws Exception {
      // ignore white space
      if (n.getNodeName().equals("#text")) {
         if (n.getNodeValue().trim().length() == 0) return;
      }
      indent(in);
      System.out.print(n.getNodeName());
      if (n.getNodeValue() != null) {
         System.out.print(": " + n.getNodeValue());
      }
      System.out.println();
      NamedNodeMap nnm = n.getAttributes();
      if (nnm != null) {
         for (int j = 0; j < nnm.getLength(); j++) {
            indent(in+1);
            Node attr = nnm.item(j);
            System.out.println(attr.getNodeName() + ": " + attr.getNodeValue());
         }
      }

      NodeList nl = n.getChildNodes();
      if (nl == null) return;
      for (int i = 0; i < nl.getLength(); i++) {
         Node child = nl.item(i);
         walkTheDoc(child, in+2);
      }
   }

}
```

Finally – New Ground

- If you are to have a web service, you need some piece of software that
  - listens on a well known port
  - determines which piece of software has to be started
  - starts up the software, quickly and safely
  - connects the external client to the service
- At its core, this is simply an http server (well, maybe without the quickly and safely part...)

Tomcat

- Tomcat is a Java Servlet container and web server from the Jakarta project of the Apache Software Foundation (http://jakarta.apache.org)
  - Provides basic static web page and cgi program support.
  - It provides a “container” to run specific types of Java classes (those that extend the HttpServlet class)
  - It isn't quite as fast or flexible as Apache for normal web services
  - It's free.
  - It's written in Java, so it is portable

Servlets

- Servlets are simple classes that implement at least one of the methods:
  - doGet or doPost for the normal Web form access
  - doPut, doDelete
  - getServeletInfo
  - init and destroy – for acquiring and releasing resources at the beginning and end of its life.
- Some examples later

Alternative to Tomcat?

- They probably exist – WebSphere, .Net, etc.
  - Tomcat stands out due to
    - its cost (free) as part of the open source movement
    - it has lots of real-time experience behind it
- Important to remember
  - The client should care less (i.e. no browser should be concerned with the http server software
  - The server program should care less (i.e. no html designer/cgi programmer should be concerned about the http server software
  - Issues are configuration... (which is plenty!!!)

Installation

- Install Java first
- Get the right media from apache.org (mostly a matter of the correct packaging – I don't think there is any “binary” code in the distribution)
- Alternative, get jwsdp from Sun.
- Install somewhere
  - By default, it listens on 8080 for incoming requests
- Explore a little – see the web interface
- It works ok on a standalone PC.

Tomcat & Webapps

- Tomcat has the concept of a web application
  - partly from the world of java web services
  - also from the world of Java Enterprise Edition (which we will not cover here)
- Most Java IDEs support this concept as well and make it relatively easy to have a develop – load – test cycle.

“Web Applications” Directory Layouts

- Base Directory – any name you please
  - html documents
  - jsp pages
  - other images/sounds/etc.
  - WEB-INF directory
    - web.xml – deployment descriptor
    - classes directory
      - various java classes
    - lib directory
      - various “jar” files

Development Cycle

- Moving that many files and directories is a huge pain
- WAR files
  - A WAR file is-a JAR file that contains the above structure
  - A JAR file is-a tar file with a slightly better toc (manifest)
- Tomcat does not need the expanded files – it can read and deal with the WAR file directly.
- So, there is only 1 file, the WAR file, to move

Example 1 – an HTML page

- Using Netbeans
  - Create a “web module”
  - Look at the web.xml file (part of the servlet spec)
    - There is a global web.xml file that is read first
  - create a web page
  - war it
  - transfer it
  - test it

Example 2 – a Java Server Page

- Again, using Netbeans
  - Use the existing “web module”
  - Create a jsp
  - Look at the web.xml file
  - war it
  - transfer it
  - test it
    - Notice the sizable delay on the first run – that is the “servlet” being compiled

Example 3 – Servlet Example

- Use Netbeans to create
- Compile and view new web.xml
- Transfer & Run
- Modify the web.xml a bit & rerun

Example 4 – A Simple Form/Servlet

- Use Netbeans to create both
- Compile
- Load
- Test

At the Brink

- The Add servlet is getting close to what we need for a “web service”
  - A simple URL access (http://dq.arm.gov:8080/test1/sum?x=3&y=9)
  - But, more complex problems will have more complex parameters, and do we really want to re-invent passing parameters for each service?
  - SOAP – and XML Wrapper will help us standardize the client access

Response

- The Sum servlet returns a response to the client, but in the form of an html file
  - fine for humans, but for machines?
  - Could you write an html parser?
- Clearly – the same problem in the other direction, so once again, SOAP and XML to the rescue.

And other problems

- Can we skip all this web stuff and just move to a (conceptual) RPC?
- How to represent a web service to the world?
- How is the service found?

Coming Up Next

- SOAP
- A Simple (real) Web Service
- WSDL
- UDDI

Homework

- Install Tomcat somewhere – even on your PC
- Look at the sample programs
- If you are a Java programmer
  - Create a database on the server and have the servlet access the database in response to a request.
- If you are a C++/Perl programmer, investigate C++/Perl solutions comparable with Tomcat.
- Write code that permits one to view log and/or configuration files for Tomcat remotely.

Annotated Bibliography

Java Web Services by Chappell & Jewell – an O'Reilly Book. Useful in conjunction with other books.

Web Services – Essentials by Cerami – an O'Reilly Book. As above – less of a Java spin, but still, plenty of Java.

Java Web Services in a Nutshell by Topley – an O'Reilly Book. Very good reference work. You need it.

Professional Java XML by Ahmed, et al. - a Wrox book. Very good collection of tutorials. Highly recommended.

Java & XML by McLaughlin – an O'Reilly book. Less expansive than the above.

Tomcat – The Definitive Guide by Brittain & Darwin – valuable reference

Bitter Java – Bruce Tate, a Manning Book – Absolutely Required Reading on Java Antipatterns